Guaranteed Road Network Search with Small Unmanned Aircraft

Michael Dille,[1,2] Ben Grocholsky,[2] and Sanjiv Singh[2]
[1]*SGT Inc., NASA Ames Research Center*
[2]*Robotics Institute, Carnegie Mellon University*

***Abstract*— The use of teams of small unmanned aircraft in real-world rapid-response missions is fast becoming a reality. One such application is search and detection of an evader in urban areas. This paper draws on results in graph-based pursuit-evasion, developing mappings from these abstractions to primitive motions that may be performed by aircraft, to produce search strategies providing guaranteed capture of road-bound targets. The first such strategy is applicable to evaders of arbitrary speed and agility, offering a conservative solution that is insensitive to motion constraints pursuers may possess. This is built upon to generate two strategies for capture of targets having a known speed bound that require searcher teams of much smaller size. The efficacy of these algorithms is demonstrated by evaluation in extensive simulation using realistic vehicle models across a spectrum of environment classes.**

I. Introduction

Small unmanned air vehicles (UAVs) are increasingly popular in surveillance, mapping, and rescue applications owing to their portability and minimal infrastructure requirements while retaining advantages over ground-based vehicles such as higher speed and lower vulnerability to obstacles. One particular benefit of their relatively low cost and ease of deployment is the enabling of cooperative missions requiring moderately large teams.

One such mission is to search for a potentially fleeing evader, particularly in an urban environment in which half the world's population lived by 2007, a proportion expected to reach 70% by 2050. [1] Beyond simple ubiquitousness, urban areas provide convenient simplifications for pursuit of targets assumed to lie within their road networks. Crucially, an evader's state space imposed by a typical road network possesses greatly increased *sparsity* (proportion of the continuous bounding region that is of relevance) and much-decreased *connectivity* (number of states connected to any other given state) relative to an equivalently-sized open area. This is due to a road network's fundamental structure as a collection of one-dimensional paths with discrete points of connectivity and effectively reduces the search space while increasing predictiveness of target motion.

In general, targets of such a search may be assumed to be uncooperative, with behaviors that may range from mere ignorance of the ongoing search to adversarial evasion in which every effort is made to avoid capture. As it is highly difficult in practice to effectively model targets and the risk of target loss may be very undesirable, conservative strategies making minimal assumptions are preferable. Additionally, performance guarantees (e.g. that a strategy is mini-max optimal in a game-theoretic sense) provide necessary confidence that searcher resources are being used effectively.

Pittsburgh, Pennsylvania, USA, 15213 (Email:mdille3@ri.cmu.edu)

This paper builds upon graph-based pursuit-evasion abstractions which, when combined with appropriate mappings to UAV capabilities, provide performance-guaranteed search behaviors and represents some of the first efforts to apply such formalisms to physical UAV teams. First, a basic time-invariant strategy is presented that offers guaranteed capture of arbitrarily agile evaders using searchers having even severe motion constraints, while further minimizing mission duration as a secondary objective. This is extended in two variations of a less-conservative approach for capture of targets having a known maximum speed whose team size requirements scale smoothly with searcher motion abilities. Extensive simulation trials using physically-based models are shown to validate the viability of these strategies.

In this work, the scale of the UAVs used is assumed to be that of several to approximately ten small, field-launched reconnaissance aircraft, however the ideas presented are equally applicable to other types. Time (and hence speed) insensitivity is intrinsic to the basic strategy, and its extensions presented primarily require larger team sizes for slower searchers. This coincides well with the probable availability of larger teams of smaller agents such as micro air vehicles and smaller teams of larger, faster aircraft. The task also maintains relevance across the vehicle scale spectrum in that for smaller aircraft, direct control of the vehicle's location is required to place a small sensor footprint, while larger aircraft with wide-area sensing still must typically direct the search attention of a foveal view via gimbal pointing or sub-window selection.

II. Related Work

Area search for mapping or target search has been an early and ongoing application of UAVs. A taxonomy of search tasks may be made based upon target type. For stationary targets (e.g. landmarks), the objective is simply to observe all relevant locations in the environment, typically in minimal time. For uniform coverage priority, classical geometric strategies include straight-line sweeps forming lawnmower, Zamboni, or box-spiral patterns [2] as well as outwardly spiraling orbits. [3] In the road network domain, aerial coverage has taken the form of implementations of classical graph coverage algorithms such as the Chinese Postman [4] or Traveling Salesman [5] problems carefully tailored to aircraft motions and constraints.

For moving targets or areas containing differing coverage priority, the search task is instead one of generating coverage

paths that minimize expected time to detection that may include repeated observation of regions recontaminated by possible target re-entry. Typical approaches entail maintaining a cellular or particle representation of search progress and the use of a motion planner to maximize detection objectives. Seminal examples include greedy search of probability grids, [6] a general multi-step planning framework for heterogeneous terrain, [7] and cooperative horizon search using predicted updates to Bayesian estimators. [8] To the limited extent it has been considered, the specific case of road networks has been typically approached by assigning probability to areas corresponding to road segments and little to areas outside the network. [8], [9]

Finally, evasive or adversarial targets possessing the specific goal of avoiding capture represent the most challenging case and must be approached in a game-theoretic manner to either provide (or refute) a capture guarantee or maximize detection probability in a mini-max sense assuming that the target will act most inconveniently for the searchers at every opportunity. Adversarial search with UAV teams has received fairly little attention, partly owing to the difficulty of operating in open environments with motion constraints that may greatly exceed the target's. Applicable strategies previously proposed include shrinking circular flight patterns [10] or coordinated sweeps [11] that constrain a bounded-speed target to increasingly smaller areas at the cost of requiring many or fast searchers. A rare instance of aerial operation within road networks uses fixed unmanned ground sensors to produce optimal visitation patterns to provide bounded (guaranteed) capture time. [12]

General pursuit-evasion has a long history, recently reviewed in the context of mobile robotics by Chung et al, [13] most classical formulations of which pertain to adversaries in continuous or polygonal planar spaces. Two applications to aerial search involve the use of discrete probabilistic search for a randomly moving evader by an air-ground team providing guaranteed finite capture time [14] and continuous differential game theory to provide multi-UAV interception of a target whose current position but not future motions are known. [15] This paper builds on the sub-field of *graph search*, representing adversaries in discrete space, well summarized in a survey by Fomin and Thilikos. [16] Two existing robotics implementations model evaders as moving between nodes for indoor search and rescue [17] or as areas to be cleared with edges weighted by the number of searchers required to guard them. [18] Strategies presented here instead use a variation modeling evaders as lying on edges, building upon a previously proposed abstract clearing algorithm. [19]

Additional aspects intrinsic to aerial search but beyond the scope of this work include target detection, tracking, and geolocation (localization), a summary of existing strategies for each of which may be found in a previous paper. [20] Location estimation of previously-detected targets within road networks has received substantial attention within the radar tracking community, classically using multi-hypothesis Kalman filters [21] given the discretely partitioned state space.

III. Problem Statement

In this problem, a target is assumed to lie wholly within the edges (road segments) of the graph underlying a physical road network, and intersections (graph nodes) are treated as infinitesimal. Targets are assumed to be adversarial in that they will make every effort to avoid capture or to maximize time to capture. The additional property of *omniscience*, or awareness of pursuer strategy and state, often allowed in pursuit-evasion, is further permitted here, though this is surely stronger than necessary for practical missions.

Two forms of target motion are considered, both conservative formulations requiring minimal target knowledge and modeling. The first is that targets may move infinitely fast in any direction within the graph, corresponding to targets about which no model is available or the case of using extremely slow aircraft. The other is that targets may move in any direction but with a maximum speed bound, which is applicable to targets about which some information is available (e.g. pedestrian vs. automobile). This permits greatly reduced conservatism in resulting strategies without requiring additional information about target intent that may influence its choice of paths.

Initially, the road graph may start with any arbitrary subset marked as contaminated (potentially containing a target). When using the infinite speed target motion model, contamination spreads throughout the entire graph immediately (with the consequence that such graphs must be finite, existing in a bounded region).

Aerial searchers are not constrained to the graph themselves and may move freely within the environment subject to any applicable kinodynamic constraints. Vehicles are not required to be of any type (e.g. rotorcraft vs. fixed-wing), however for simplicity and generality, a fixed-wing motion model is assumed, as it represents a minimal capability emulatable by others. For simulation purposes, a simple but widely-adopted constant-altitude planar Dubins vehicle [22] is used, capturing requirements for a minimum forward speed and minimum turning radius. For greater realism, a coordinated-turn constraint is also added, dictating that the vehicle must bank (roll) to turn as a function of steering angle and turn radius.

No specific requirements for searcher sensing modality are needed beyond possessing a presumed field of view and that target (non-)presence must be detected. Given payload limitations of small aircraft, the minimal sensor assumed is a camera accompanied by an unspecified detection algorithm. The location of its footprint (intersection of the viewable frustrum with the ground) depends on mounting, which is most commonly either forward-pointing, side-pointing, or gimbaled. For maximum applicability and relevance to widely-fielded aircraft, a fixed side-pointing camera is used here. The size of the footprint need only be sufficient to view the entire width of a road segment and has width approximately equal to the minimum orbit radius for physical camera parameters selected for simulations here. The only critical sensing assumption is that if a UAV's field of view

passes over the location of a target, then it is detected with certainty (false positives are considered irrelevant). In practice, a stochastic model requiring minimum dwell time or repeated overflight may be necessary to approach this. Finally, accurate sensor pointing is necessary to avoid incorrectly marking areas as cleared. Substantial robustness to vehicle state error that would preclude this is incorporated by marking only a small area near the intended center of the field of view as clear, which may be safely assumed to lie within the larger true footprint.

Overall, the ultimate goal of any search is to guarantee eventual detection of any targets that may be present, while minimizing the number of searcher agents required as a presumed scarce resource. Subject to this, a strong secondary objective is to minimize overall mission time.

IV. Search for Infinite-speed Targets

A. Basic Strategy

For adversarial targets without a speed bound, the underlying algorithm used is an abstract graph search for undirected graphs and edge-occupying evaders previously presented by Barrière et al. [19] for the special case of trees. Intuitively, this is fundamentally a depth-first traversal in which edges are swept during descents and any node having more than one child is guarded during child searches lest contamination from unsearched subtrees return to previously swept subtrees via that node. This is summarized formally in Algorithm 1, which returns the *search number* (number of agents required) and a *search schedule* (ordering of actions) comprising a sequence of two primitive actions: *guarding* of a node and *sweeping* of an edge.

```
search_schedule = [ ]
Function clear_tree(root, parent=∅, preguarded = [ ])
begin
    search_num = 0
    if parent ≠ ∅ then
        search_schedule.append(SWEEP[parent,root])
        search_num = 1
    if |root.children| > 1 AND ¬preguarded[root] then
        search_schedule.append(BEGIN_GUARD[root])
    foreach child ∈ root.children do
        search_num =
        MAX(search_num,clear_tree(child,
        root,guarded))
    if |root.children| > 1 AND ¬preguarded[root] then
        search_schedule.append(STOP_GUARD[root])
        search_num = search_num+1
    return search_num
end
```

Algorithm 1: Tree search for infinite-speed targets

Of course, physical environments rarely form convenient trees and instead contain cycles. Unfortunately, even computing the search number of general graphs is intractable. [23] Instead, an intermediate solution similar to that proposed by Hollinger [17] is proposed. Specifically, a search schedule is generated in an anytime fashion by iterating over randomized (edge-)spanning trees of the entire graph, computing the search number for each as the that of the tree plus the number of additional guards required to break all cycles by placing a guard at one end of each edge discarded by that spanning tree. This is summarized in Algorithm 2. Given this algorithm, subsequent strategies are defined in terms of tree graphs and may be assumed to be wrapped by this. Required team size growth with increasing numbers of cycles is evaluated in Section VI.

```
guard_nodes = [ ]
search_schedule = [ ]
Function search_number = clear_graph(graph, rooti)
begin
    best_search_schedule = [ ]
    best_searchnum = ∞
    while time remains do
        // e.g. random Kruskal's
        tree = get_random_spanning_tree(graph)
        curr_guard_nodes = [ ]
        foreach edge ∈ graph do
            if edge ∉ tree then
                i = rand({0,1})
                nexus = tree.nodes[edge.endpoint[i]]
                parent = tree.nodes[edge.endpoint[1-i]]
                new_leaf = duplicate_node(nexus)
                tree.insert_node(new_leaf)
                tree.insert_edge(parent, new_leaf)
                curr_guard_nodes.insert(nexus)
        // Algorithm 1
        searchnum = clear_tree(tree.node[rooti], ∅,
        curr_guard_nodes)
        if searchnum+curr_guard_nodes.size <
        best_searchnum+guard_nodes.size then
            best_search_schedule = search_schedule
            best_searchnum = searchnum
            guard_nodes = curr_guard_nodes
    search_schedule = best_search_schedule
    return best_searchnum
end
```

Algorithm 2: Anytime search for general graphs

B. Application to UAVs

Implementation of the aforementioned abstract search with UAV teams requires several extensions. First, the primitive actions *guard* and *sweep* must be mapped to executable behaviors. To perform a guard action, a UAV (or an assemblage of several) must be able to loiter over an area at least as large as an intersection within the environment and provide consistent viewing of it. This is directly applicable to hovering rotorcraft or fixed-wing aircraft with either gimbaled or fixed side-pointing cameras via a persistent orbit. Fixed-wing aircraft with downward or fixed-angled cameras pointing along the forward axis (e.g. downward or ahead) are more challenging and must be able to either loiter (e.g. fly in a tight figure-eight pattern) while maintaining a sufficiently large viewing area or join others in doing so such that the union of their sensor footprints covers the guard point at all

times. Likewise, to perform a sweep action, a UAV must be able to move its field of view to follow the entire path of an edge. While clearly straightforward for simple line segments, edges whose path contains complex curvatures cannot in general be followed by vehicles with turning-radius constraints and may be treated as containing intermediate nodes so that the curvature of any single edge does not exceed these constraints. These may then be followed by either orbiting around these intermediate points (for side-facing and gimbaled sensors) or by pairing with an additional agent to guard these points during repositioning.

Next, search schedule effort must be distributed within a team. For the ground-based case, effort assignment is a non-issue as at each step all free searchers may be conceptually thought of as executing the next action in tandem, leaving any available agent behind as a guard where required. In contrast, aerial searchers are not themselves constrained to the graph and may move between locations arbitrarily (subject to motion constraints), permitting substantial mission time optimization while complicating task assignment. In this case, it is beneficial to perceive the underlying tree being traversed as a dependency graph dictating a topological hierarchy capturing which nodes must be visited before others, within which searchers are in fact free to choose the sequence in which graph elements are actually visited. For guard actions, a minimal-length trajectory between initial and terminal locations need simply be chosen. However, sweep actions admit greater freedom in that subtree ordering and (where possible, for edges leading to leaf nodes) edge sweep direction may be chosen to minimize intermediate maneuvering. Globally, given a homogenous team, assignments and transitions between guard and sweep roles may be chosen to minimize flight distance given instantaneous poses at the time of transition.

These freedoms may be formalized by defining two explicit extensions to abstract search schedule generation. First is *sweeper pre-positioning*. Conceptually, a sweeper dependent on a guard must wait for the guard to begin its loiter before the sweep can begin. Rather than naively waiting and producing a lengthy idle period between the activation of a sweeper and the start of the actual sweep as it maneuvers into position, respective maneuvering times for an en-route guard and its dependent sweeper may be estimated, and the sweeper may prematurely begin its maneuver in parallel towards the target edge in anticipation of the guard's arrival before it.

The other extension is to apply *action optimization* acknowledging that subject to the topology of the dependency tree, edges may be swept by any available agent, in any order, and (when permissible) in either direction. This produces an assignment, sequencing, and direction-choice optimization problem that may be solved to minimize total mission time. Unfortunately, this represents a large constrained combinatorial optimization and is intractable to solve exactly for any substantially-sized environment, and an approximate approach is applied. The simplifying assumption is made that optimal assignment and action sequencing within any subtree is optimal globally, permitting independent single-step optimization within each subtree. Specifically, subtrees of a given node are searched recursively, independently, in increasing order of search number, and action optimization is only applied to leaf subtrees (comprised of only a path). Given a guard at a parent node, all leaf subtrees immediately below it may be treated as a basket of paths to search, whose assignment, sequence, and search direction are free variables. This is solved optimally using trivial computation effort at each encounter using a Traveling Salesman based coverage algorithm developed in a prior publication. [5]

The overall algorithm is summarized formally in Algorithm 3 and is chosen as a reasonable balance of complexity and approximation in that it is readily implemented while capturing key extensions. Two steps during its execution on a small demonstration environment are shown in Figure 1. In this example, two UAVs (whose fields of view from left-facing cameras are shown as trapezoids) clear an initially fully contaminated (marked as red) environment. Shown are the first two steps in which a guard is emplaced (A), after which the other agent sweeps leaf two subtrees (B and C) optimally. No longer needed, the guard transitions to a sweep of the larger subtree (D), during which the other agent begins a preemptive inward sweep of the next leaf subtree (E), reaching its parent (F) strictly after it too will be guarded.

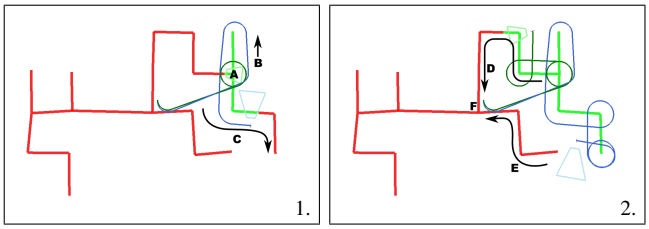

Fig. 1: Two steps in search of small demonstration environment for an infinite-speed evader

V. Search for Bounded-speed Targets

In practice, infinite-speed search can be highly conservative and resource-intensive, requiring many agents (especially in highly cyclic environments) and long search times. Searching a large area contaminated by an evader presumed to diffuse instantaneously is unnecessary with additional information, such as a rough location and some speed bound, in which case the search may be simplified. Example scenarios include a sighting reported at a remote location or the need to search a small portion of an unbounded area with connectivity to effectively arbitrarily distant locations. These may be treated as initial contamination of some small map subset that diffuses at a worst-case target speed.

Intuitively, infinite-speed search as proposed may be built upon to produce effective search strategies for bounded-speed targets. If, for instance, an infinite-speed search of an area can be performed before the initial diffusion can exceed the extent of that area, then capture is guaranteed. Alternatively, if diffusion can be stemmed by guarding appropriate boundary points, then a search may be completed at leisure of the now-bounded area. These two examples inspire the following two respective strategies. The first is best suited to smaller teams of fast agents, while the second

```
Function [search_schedule,search_number] =
clear_tree_optimized(root, parent=∅)
begin
    schedule = [ ]
    search_num = 0
    if parent ≠ ∅ then
        schedule.append(SWEEP[parent,root])
        search_num = 1
    if |root.children| > 1 then
        schedule.append(BEGIN_GUARD[root])
    foreach child ∈ root.children do
        [child_schedule[i],child_search_num[i]] =
        clear_tree_optimized(child, root))
    search_num =
    MAX(child_search_num[...],search_num)
    child_order = sort([1...|root.children|] by
    child_search_num)
    leaf_subtrees = find(children[child_order] == 1)
    nonleaf_subtrees = find(children[child_order] > 1)
    // Coverage optimization from [5]
    schedule =
    [OPTIMIZE_COVERAGE(children[leaf_subtrees])]
    foreach child_index ∈ nonleaf_subtrees except last
    do
        schedule =
        [schedule,child_schedule[child_index]]
    if |root.children| > 1 then
        schedule.append(STOP_GUARD[root])
        search_num = search_num+1
    if nonleaf_subtrees.size > 0 then
        schedule =
        [schedule,child_schedule[nonleaf_subtrees.last]]
    return [schedule,search_num]
end
```

Algorithm 3: Leaf-optimizing tree search

is tolerant of slower agents as long as more are available. Both more realistically assume an available team size, which need never be larger than an area's search number.

A. Live Search

Seeded by an initial partial contamination and guided by a bound on its growth rate, the graph subset of interest may be seen as a time-varying graph $\mathbf{G}'(t) \subseteq \mathbf{G}$, where $\mathbf{G}$ is the entire surrounding map. If at any time this predicted graph subset can be searched by any guaranteed means (such as the proposed infinite-speed search) within this time t, then subject to the validity of the speed bound, guaranteed capture is assured. This notion is phrased formally as Algorithm 4, which alternates between predicting map growth and estimating its search time.

Starting at $t_0 = 0$, the search time t_1 of the graph $\mathbf{G}_0 = \mathbf{G}(t_0 = 0)$ is estimated. If $t_1 \leq t_0 = 0$ (unlikely), then success is declared. Otherwise, map growth is predicted up to t_1 as $\mathbf{G}_1 = \mathbf{G}(t_1)$, the search time t_2 of $\mathbf{G}_1$ is estimated, success declared if $t_2 \leq t_1$, and iteration continued if not. Failure is declared if at any iteration the search number of $\mathbf{G}_i$ exceeds the available team size. Note that large jumps in the growth of $\mathbf{G}$ are typical, resulting in few iterations before termination.

```
search_schedule = [ ]
Function [search_number,duration] =
bounded_recaptureA(tree, max_target_speed,
contaminated_edges[ ], agent_poses[ ])
begin
    new_search_time = 0
    repeat
        search_time = new_search_time
        tree_subset = diffuse_contamination(tree,
        contaminated_edges, max_target_speed,
        search_time)
        // Algorithm 1
        search_num = clear_tree(tree_subset)
        if search_num > |agent_poses| then
            return FAILURE
        new_search_time =
        estimate_search_duration(tree_subset,
        agent_poses)
    until new_search_time ≤ search_time ;
    return [search_num, new_search_time]
end
```

Algorithm 4: "Live search" for bounded-speed targets

A critical element is the estimation of a graph subset's search time, admissibility of which requires merely a conservative estimate. This may be done by any means available, such as internal forward simulation of vehicle behavior. As this may be time-consuming, a simpler conservative estimate using easily-computed worst-case maneuvering times is compared in Section VI.

A trivial execution example on the same small demonstration environment is provided in Figure 2. Here, a remote target sighting is reported at a location away from a loitering search team. It is quickly computed that a single searcher is sufficient to clear the resulting graph subset if the search is begun at sufficient back-off from the initial sighting (A), and one agent is dispatched to perform this sweep (B).

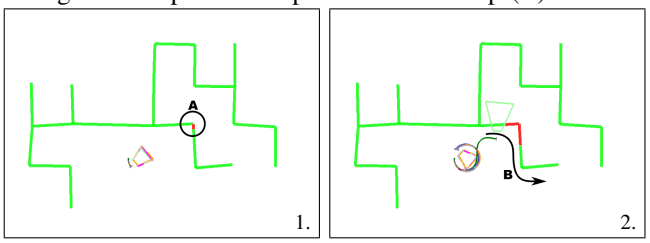

Fig. 2: Two steps in "live search" of small demonstration environment for a bounded-speed evader

B. Bound and Search

Alternatively, one might attempt to block the growth in map contamination, at which point a fixed bounded graph is contained within and may be searched by any guaranteed search strategy, regardless of search duration. In practice, this corresponds to guarding of strategic boundary nodes before contamination reaches them. Similar in logic to the preceding

method, given a predicted graph subset $\mathbf{G}(t)$ corresponding to the contaminated set at time t and provided all bounding nodes can be reached within time t, then capture is assured if sufficient agents are available to simultaneously guard these nodes and perform the interior search.

Phrased formally as Algorithm 5, the contaminated subgraph $\mathbf{G}_0 = \mathbf{G}(t_0 = 0)$ at time t_0 is computed, along with all boundary nodes linking nodes in $\mathbf{G}_0$ to nodes in the original surrounding graph. If sufficient agents are available and these can all be reached and guarding begun by time t_0 (unlikely), then the search number of the internal subgraph is also computed. If the total number of boundary guards plus internal searchers required is at most the total number of available agents, then success is declared. Otherwise, the algorithm iterates to an incremental t_1 corresponding to a minimal discrete growth in the contaminated graph and the same tests performed. Eventually, the number of agents required to perform the internal search will exceed that available (failure), or the overall graph extent will be reached (if bounded), falling back to the equivalent of search of the original graph.

```
search_schedule = [ ]
Function [search_number,guard_assignment] =
bounded_recaptureB(tree, max_target_speed,
contaminated_edges[ ], agent_poses[ ])
begin
    growth_time = 0
    tree_subset = [ ]
    repeat
        [tree_subset,distance_ext] =
        diffuse_one_edge(tree, tree_subset)
        growth_time = growth_time + distance_ext /
        max_target_speed

        boundary_nodes = get_boundary_nodes(tree,
        tree_subset)
        if |boundary_nodes| ≥ |agent_poses| then
            continue

        [guard_time,guard_assignment] =
        get_node_reach_time(boundary_nodes,
        agent_poses)
        if guard_time ≥ growth_time then
            continue

        // Algorithm 1
        search_num = clear_tree(tree_subset, ∅,
        boundary_nodes)
        if search_num > |agent_poses| then
            return FAILURE
    until search_num + |boundary_nodes| ≤
    |agent_poses| ;
    return search_num
end
```

Algorithm 5: "Bound-and-search" for bounded-speed targets

A simple optimization applied is to make use of boundary guards for the interior search. While maintaining guards at a location only as long as necessary and re-introducing them to the search team thereafter produces complicated interactions requiring computationally-intensive optimization, two simple steps may be taken. First, guards may at least be treated as permanently guarding a given node, avoiding temporary guards if it has multiple subtrees. Further, boundary guards lying on leaf nodes in the contaminated subgraph can sweep up the path on which they lie until a node is reached having more than one child. This time-invariant step may substantially shrink the size of the interior search graph, reducing its search number.

A small example of executing this approach is shown in Figure 3. In this example, a delayed remote sighting report near an intersection allows substantial growth in contamination before searchers can respond (A). Three are dispatched to stem growth in contamination along incident edges, which after an inward sweep of each (B, C, and D) leaves an empty interior graph requiring no further searchers.

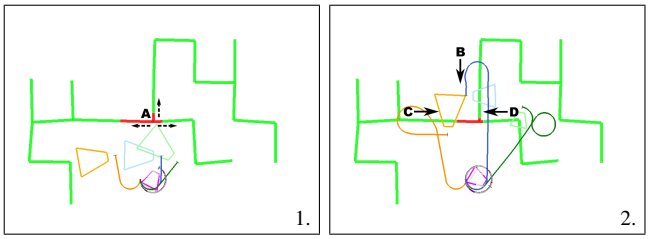

Fig. 3: Two steps in "bound and search" of small demonstration environment for a bounded-speed evader

VI. Experimental Results

For evaluation and comparison, repeated realistic simulations on varying maps with varying-sized teams was performed. UAV teams were assumed to be comprised of fixed-wing aircraft approximated by a Dubins vehicle using coordinated turns with fixed, side-angled cameras with parameters similar to that of widely-fielded small reconnaissance UAVs. In these simulations, a full implementation including injected control noise, state error, feedback control, and camera projection for field of view estimation is utilized to capture important real-world effects.

Test environments were selected from a spectrum of randomly-generated maps formed from perturbed Manhattan grid subsets intended to realistically mimic physical road networks, parametrized by overall size, fraction of the entire grid present, and block size (edge length). Roughly, a complete grid of specified block size from 100m (dense urban) to 1km (sparse rural) is generated, junction locations are perturbed to increase required maneuvering, and a specified fraction (from 20% to 70%) of edges are removed, the remaining fraction being labeled the block density. Clearly, true evaluation is best performed on samples of real-world areas, but this method permits rapid high-volume testing on a wide diversity of environments. In each simulation trial, a team of searchers is launched as a group from a randomly selected point within the environment. Each data point presented is the average of several hundred trials using differing random seeds for environment generation and start location.

For thorough evaluation, simulation across both a spectrum of varying environment size and density was performed. Only selected representative results are provided here. For additional results, the interested reader is referred to an extended technical report on this work. [24]

Typical results for infinite-speed guaranteed search are shown in Figure 4. In this comparison, several algorithms are tested on varying-density (block length) maps of the same size: a road-constrained ground-based team executing Algorithm 1, a UAV team performing the same search schedule (an unoptimized search), a UAV team using leaf-optimizing tree search (Algorithm 3), and a team of idealized rotorcraft (lacking any kinodynamic constraints and hence requiring no orientation-dependent path optimization) following the same optimized strategy. As this plot shows, the proposed leaf-optimizing strategy eliminates at least 1/4 of the mission duration difference between the baseline UAV strategy and that of an ideal ground vehicle, which is itself outperformed by an ideal rotorcraft. It is worth further noting that robotic ground vehicles often move much more slowly than aircraft rather than at the same speed as assumed here, and so in practice the colored traces may lie much lower.

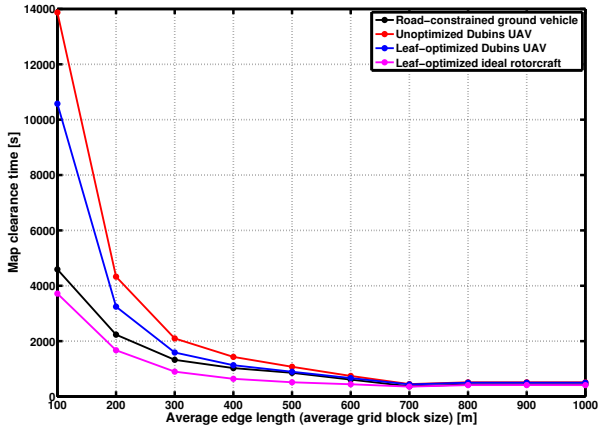

Fig. 4: Mission duration comparison for infinite-speed search

Next, the impact of graph cycles on required team size is considered. Algorithm 2 was applied to environments of varying edge density and block length, with typical examples across the density spectrum shown in Figure 5 with an edge density of 80%, considered somewhat more dense than would be expected in real-world environments and hence representing largely worst-case instances. A plot of required team size for this 80% edge density on samples of suburban (450m block length) areas of varying size is given in Figure 6. Comparison is provided against two simple guaranteed search alternatives: an open area search in which agents sufficient to fill the width of the environment sweep in parallel across its longer dimension and a Manhattan grid sweep in which one agent per block sweeps across the longer dimension in parallel, pausing at intersections while an additional agent sweeps up and down along the orthogonal axis. As this plot shows, cycle-breaking guards dominate team makeup (the search subset growing only very slowly), yet graph search is superior to these alternatives for up to moderately-sized (over 3km^2) areas, at which point required team size still remains realistic.

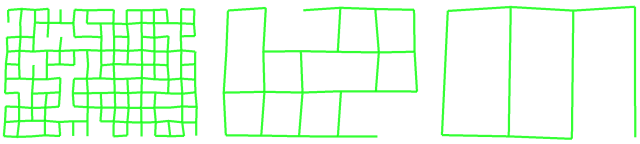

Fig. 5: Examples of test environments having 80% edge density. From left to right, these have block length 200m (urban), 450m (suburban), and 700m (rural).

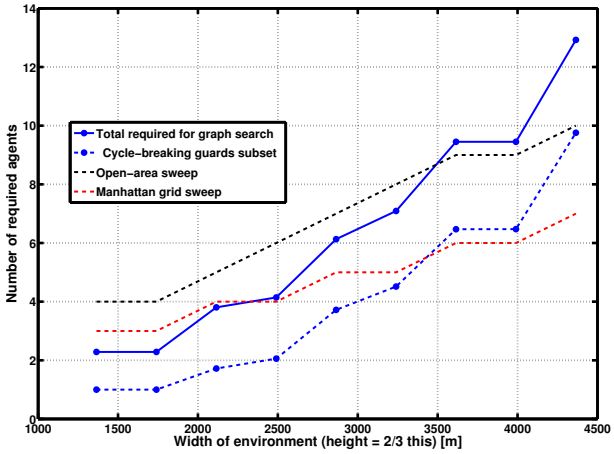

Fig. 6: Required team size for suburban (450m block length) and 80% edge density environments of increasing size

Finally, the proposed bounded-speed search strategies are similarly evaluated. Figure 7 depicts the effect of pursuer-to-target speed ratio on required searcher team size. In this case, a 3km $\times$ 2km environment of 450m (suburban) edge length and 40% edge density (so as to generate trees) is considered. For low speed ratios, the required team size approaches that needed for an infinite-speed target as expected. At higher speed ratios, this decreases, with bound-and-search requiring fewer at first and then live search requiring fewest at still higher ratios. This confirms the intuition that live search is more appropriate for smaller, faster teams. Two traces for live search are provided, corresponding to two methods for predicting search duration: sub-realtime forward simulation and a rapidly-computed conservative estimate assuming worst-case maneuvering times for every motion. The small difference in performance suggests the reasonableness of such an approximation. Likewise, average search duration for the same trials is shown in Figure 8, which carries similar overall trends except that the cross-over between methods occurs earlier and that the performance difference between the search prediction methods is greater. This implies that appropriate choices of methods may differ depending on whether minimizing required team size or mission duration is of greater importance.

VII. Conclusion

By building on existing ideas in abstract pursuit-evasion, a guaranteed target search strategy for adversarial evaders of arbitrary agility was developed for aerial searchers by mapping abstract search primitives to motions executable by real aircraft. Extensive simulation verified both the practicality

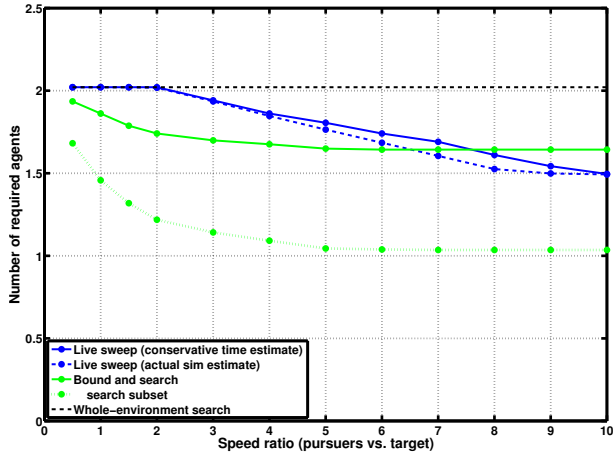

Fig. 7: Required team size vs. searcher-to-target speed ratio for bounded-speed target search in a suburban environment

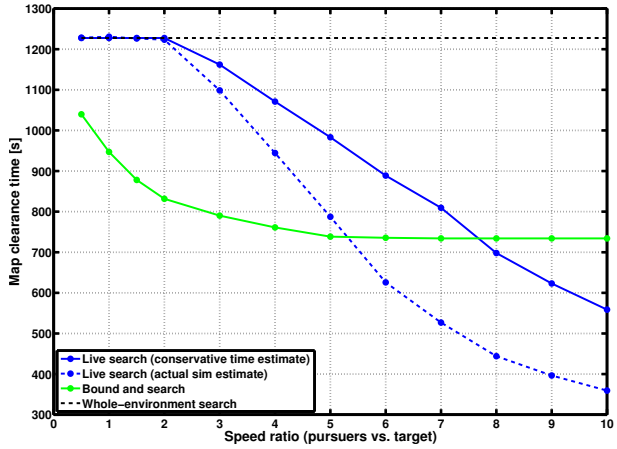

Fig. 8: Mission duration vs. searcher-to-target speed ratio for bounded-speed target search in a suburban environment

of this strategy in terms of required team size and mission duration as well as in comparison to alternative guaranteed search methods. This was extended by two variations to search for targets having known speed bound requiring much smaller team sizes and permitting operation in unbounded environments.

Many avenues for further exploration exist, including comparison to still other guaranteed search algorithms proposed for open areas and further improvement to the proposed leaf-subtree optimization by increasing this to a horizon optimization or introducing subtree search parallelism. As the full capabilities of UAVs—such as the ability to see potentially large areas including many environment elements at once—are not fully utilized and substantial conservatism in search behavior remains, alternative pursuit-evasion abstractions and UAV motion representations should be explored. Two examples include discrete formulations such as so-called cops-and-robbers and node-based graph search that might be implemented on a configuration lattice imposed on the environment.

VIII. Acknowledgements

The authors wish to thank Drs. Thanasis Kehagias, Maxim Likhachev, and Paul Scerri for thoughtful conversation and many constructive criticisms.

References

[1] The United Nations Population Fund, "UN state of the world population," 2007. [Online]. Available: http://web.unfpa.org/swp/2007/english/chapter_1/urbanization.html

[2] V. Ablavsky and M. Snorrason, "Optimal search for a moving target: A geometric approach," in *AIAA Conference on Guidance, Navigation, and Control*, 2000.

[3] M. Quigley, B. Barber, S. Griffiths, and M. A. Goodrich, "Towards real-world searching with fixed-wing mini-uavs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[4] H. Oh, S. Kim, A. Tsourdos, and B. White, "Cooperative road-network search planning of multiple UAVs using Dubins paths," in *AIAA Conference on Guidance, Navigation, and Control*, 2011.

[5] M. Dille and S. Singh, "Efficient aerial coverage search in road networks," in *AIAA Conference on Guidance, Navigation, and Control*, August 2013.

[6] F. Bourgault, T. Furukawa, and H. Durrant-Whyte, "Coordinated decentralized search for a lost target in a Bayesian world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[7] M. M. Polycarpou, Y. Yang, and K. M. Passino, "A cooperative search framework for distributed agents," in *IEEE International Symposium on Intelligent Control (ISIC)*, 2001.

[8] J. Tisdale, Z. Kim, and J. K. Hedrick, "An autonomous system for cooperative search and localization using unmanned vehicles," in *AIAA Conference on Guidance, Navigation, and Control*, 2008.

[9] C. Geyer, "Active target search from UAVs in urban environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[10] T. G. McGee and J. K. Hedrick, "Guaranteed strategies to search for mobile evaders in the plane," in *American Control Conference*, 2006.

[11] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using UAV swarms," in *ACM Symposium on Applied Computing*, 2004.

[12] H. Chen, K. Krishnamoorthy, W. Zhang, and D. Casbeer, "Continuous-time intruder isolation using unattended ground sensors on a general graph," in *American Control Conference*, 2014.

[13] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[14] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 662–669, 2002.

[15] J. M. Reimann, "Using multiplayer differential game theory to derive efficient pursuit-evasion strategies for unmanned aerial vehicles," Ph.D. dissertation, Georgia Institute of Technology, 2007.

[16] F. V. Fomin and D. M. Thilikos, "An annotated bibliography on guaranteed graph searching," *Theoretical Computer Science*, vol. 399, no. 3, pp. 236–245, June 2008.

[17] G. A. Hollinger, "Search in the physical world," Ph.D. dissertation, Carnegie Mellon University, 2010.

[18] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 32–47, 2010.

[19] L. Barrière, P. Fraigniaud, N. Santoro, and D. M. Thilikos, "Searching is not jumping," in *Workshop on Graph Theoretic Concepts in Computer Science*, 2003, pp. 34–45.

[20] M. Dille, B. Grocholsky, and S. Singh, "Persistent visual tracking and accurate geo-location of moving ground targets by small air vehicles," in *AIAA Infotech@Aerospace Conference*, March 2011.

[21] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, and I. Kadar, "Ground target tracking with variable structure IMM estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 26–46, January 2000.

[22] L. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, pp. 497–516, 1957.

[23] A. S. LaPaugh, "Recontamination does not help to search a graph," *Journal of the Association for Computing Machinery*, vol. 40, no. 2, pp. 224–245, 1993.

[24] M. Dille, "Search and pursuit with unmanned aerial vehicles in road networks," Ph.D. dissertation, Carnegie Mellon University, 2013.